

Eclipse Plug-in Development Research Project

Report 13



UNIVERSITY AT BUFFALO

September 29, 2015
Authored by: Chern Yee Chua

Eclipse Plug-in Development Research Project

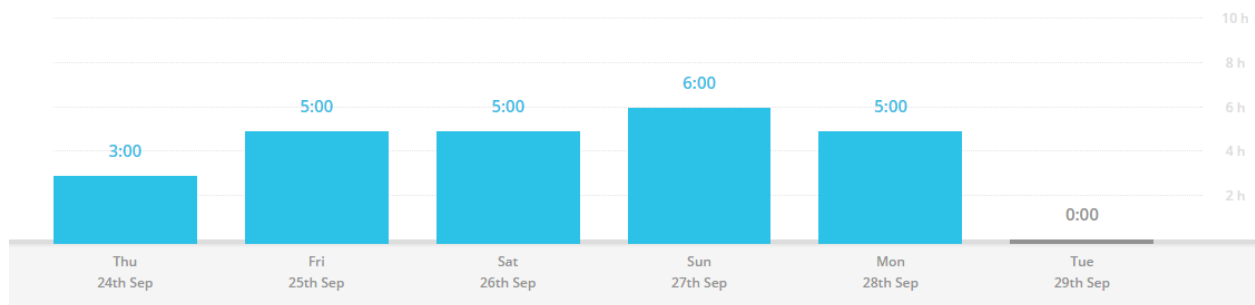
Report 13

Introduction

This is a rather huge update since last report. I have moderately changed the data output as well as implemented new features after viewing the BlackBox database.

Synopsis

I fill in 20 hours into previous two weeks (from 9/10/2015 to 9/23/2015) since I wasn't working much around that period of time.



Eclipse Plugin	24:00:00
Add original source line for better output (finally)	6:00:00
Add the error detection and fix the unintended of add and remove lines	5:00:00
Analyze the database of BlackBox	5:00:00
Change the data output style	5:00:00
Testing my own code	3:00:00

Content

There are some major updates since last report. After viewing and analyzing the data from the BlackBox (I have email you about the database they have), I realized there are certain things that I can improve to make the plugin more responsive and easier to view the data file. Plus, I tried writing code on my own and see the output file – it is really messy and very hard to interpret. Hence, I decided to make some changes to the plugin.

➤ Add the error detection functionality

```
[ERROR DETECTED(vbn)]: 32 * Syntax error on token "void", volatile expected
[ERROR DETECTED(vbn)]: 32 * Syntax error, insert ";" to complete FieldDeclaration
[ERROR DETECTED(vbn)]: 32 * void is an invalid type for the variable m
[ERROR DETECTED(vbn)]: 30 * Syntax error, insert "}" to complete ClassBody
[ERROR DETECTED(vbn)]: 32 * Syntax error on token ")", { expected after this token
[ERROR DETECTED(vbn)]: 80 * Syntax error, insert "}" to complete ClassBody
```

I have added the error detection function into the plugin. Whenever there is error in the editor, it will print out the error message instead of going on. The error message is actually the same as those little markers on side bar of the editor. The other benefit is that it fixes the previous issue that I have – whenever there is syntax error in the editor, some of the output lines will be accidentally removed and that complicates the output data. With this new implementation, I would say it is killing two birds with one stone!

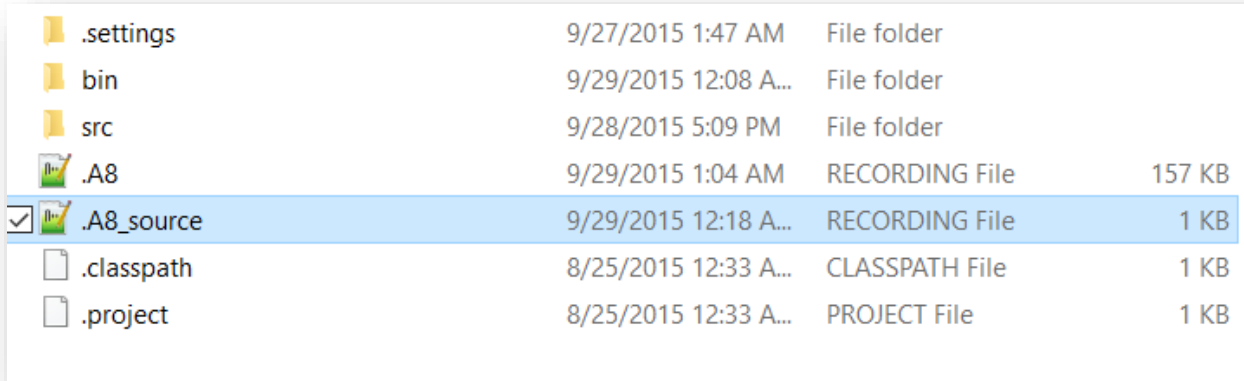
➤ Get source lines from the editor








```
[INSERT]: else {
[INSERT]: b=30;
[INSERT]: }
[LINE ADDED (vbn)]: 40 $ [ASSIGNMENT] (b) | operator (=) | token (30) |
[LINE ADDED (vbn)]: 40 $ CLOSE [ELSE_STATEMENT]
[DELETE]: else {
[DELETE]: b=30;
[DELETE]: }
[LINE REMOVED (vbn)]: 40 $ [ASSIGNMENT] (b) | operator (=) | token (30) |
```

The lack of useful sources and libraries makes the whole retrieving data from the CompilationUnit so much difficult. Due to the incomplete and complexity of ASTNode class, there is no way for me to retrieve the source line from editor directly, that's why I came out with visiting AST nodes at the first place. It looks uglier and harder to interpret, also some of the elements that are not belonged to ASTNode such as null value, else block and etc. We would definitely miss capturing some data.

Luckily, I found an algorithm that takes care of the situation and with some twerking and changing of the code, I manage to get the source lines of changes from the editor. With two ways of recording data, it ensures the data is accurate.

➤ Change data output style



	.settings	9/27/2015 1:47 AM	File folder	
	bin	9/29/2015 12:08 A...	File folder	
	src	9/28/2015 5:09 PM	File folder	
	.A8	9/29/2015 1:04 AM	RECORDING File	157 KB
<input checked="" type="checkbox"/> 	.A8_source	9/29/2015 12:18 A...	RECORDING File	1 KB
	.classpath	8/25/2015 12:33 A...	CLASSPATH File	1 KB
	.project	8/25/2015 12:33 A...	PROJECT File	1 KB

I move the initialized source file part to another recording file for a better view.

Conclusion

The BlackBox actually has a data tables for method invocation, console result and compilation error detection. I think those are pretty cool stuff but I doubt the implementation would be any easier. Also, it is unnecessary since we can see the code from students and that shouldn't be something that we are concerned for. The rest of the functions of BlackBox (the one that we need) are pretty much the same as our plugin, such as opening / closing project, changing packages and so on so for.

Up to this point, I can't think of anything to include or change. Hopefully we will get the green light to record data very soon.

Thank you for your time!